WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, May 27, 2005

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L11	l9 and (control adj2 processor)	1
☐	L10	(writable adj3 memory) and l9	0
☐	L9	((integrated adj2 circuit) or IC) same(plurality adj3 processor) same((data or media) adj2 stream)	10
☐	L8	((integrated circuit) or IC) near8 (plurality processor) near8 ((data or media) stream)	0
☐	L7	l5 and ((integrated circuit) or IC)	4
☐	L6	l5 and ((integrated cirduit) or IC)	0
☐	L5	(plurality adj3 processor) same((data or media) adj2 stream) same (control adj2 processor) same instruction	7
☐	L4	((integrated adj2 cirduit) or IC) same(plurality adj3 processor) same((data or media) adj2 stream)	2
☐	L3	((integrated cirduit) or IC) same(plurality processor) same((data or media) stream)	0
☐	L2	((integrated cirduit) or IC) near8 (plurality processor) near8 ((data or media) stream)	0
	DB=USPT; PLUR=YES; OP=ADJ		
☐	L1	5794060.pn.	1

END OF SEARCH HISTORY

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L4: Entry 2 of 2 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5872988 A
TITLE: Parallel data processing device having a concatenated data path between elementary processors

Detailed Description Text (4):
FIG. 2 shows the architecture of a part of a data processing device in accordance with the invention, said architecture emphasizing interactions between the data streams implemented in and/or by elementary processors operating in the SIMD mode. The plurality of elementary processors EP1-EPn receive at their input instructions from a control bus IC and data from a data bus ID. Each elementary processor delivers a result to an individual output bus OB1-OBn, via output means 17.sub.1 - 17.sub.n. In order to ensure that each elementary processor can communicate with its nearest neighbours, each elementary processor comprises communication means 15.sub.1 -15.sub.n enabling the transfer of data via a concatenated path DP. It is thus possible to transfer data gradually to the other elementary processors. These data exchanges may concern any data present in the elementary processor. The data streams existing in each elementary processor, therefore, are no longer independent. This is particularly useful when the contribution made by an elementary processor is associated with the contributions made by the other processors. This is the case, for example when the plurality of elementary processors EP1-EPn constitutes a vectorial processor VP delivering a plurality of results constituting a vectorial result. The latter is then generally processed by a vector-scalar unit VSU which transforms the vectorial type of result into a scalar type of result.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L5: Entry 5 of 7 File: USPT Nov 14, 2000

DOCUMENT-IDENTIFIER: US RE36954 E
TITLE: SIMD system having logic units arranged in stages of tree structure and operation of stages controlled through respective control registers

CLAIMS:

1. A parallel computer system using a single instruction stream multiple data stream (SIMD) method, said parallel computer system having a controller and a plurality of processor elements, each of the processor elements having storage means for storing data to be processed and outputting output data processed by the processor elements, the controller controls operation of the processor elements, and said parallel computer system performing processing of the data based on a calculation control signal and a synchronization signal transmitted from the controller, said parallel computer system comprising:

data collection means, connected between the processor elements and the controller and arranged in a .[.binary.]. tree configuration having stages, for receiving the output data from the processor elements responsive to the synchronization signal generated by and received from the controller, for performing a predetermined calculation based on the stages in the .[.binary.]. tree configuration, and for outputting calculated data to the controller; and

calculation control means, connected between said data collection means and the controller and arranged in series corresponding to each of the stages, for transmitting the calculation control signal from the controller to said data collection means based upon a pipe-line method to perform the predetermined calculation in said data collection means,

wherein said data collection means comprises a plurality of gathering logic units connected to each other in the .[.binary.]. tree configuration having the stages, first gathering logic units of a first stage of the stages receives the output data from each of the processor elements and outputs first calculation data, second gathering logic units of a second stage of the stages receives the first calculation data obtained from the first stage, and the first calculation data obtained from the second stage is output to a final gathering logic unit of a first stage of the stages as second calculation data, the final calculation data obtained from the final gathering logic unit of the final stage responsive to said second calculation data is output to the controller, and

wherein said calculation control means comprises a plurality of control registers each corresponding to one of the stages, each of the plurality of control registers connected in series to each other by the pipe-line method, and each of the plurality of control registers sequentially outputting the calculation control signal to each of the gathering logic units in the corresponding stage.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L5: Entry 7 of 7 File: USPT Jul 20, 1993

DOCUMENT-IDENTIFIER: US 5230057 A
** See image for Certificate of Correction **
TITLE: SIMD system having logic units arranged in stages of tree structure and operation of stages controlled through respective control registers

CLAIMS:

1. A parallel computer system using a single instruction stream multiple data stream (SIMD) method, said parallel computer system having a controller and a plurality of processor elements, each of the processor elements having storage means for storing data to be processed and outputting output data processed by the processor elements the controller controls operation of the processor elements, and said parallel computer system performing processing of the data based on a calculation control signal and a synchronization signal transmitted from the controller, said parallel computer system comprising:

data collection means, connected between the processor elements and the controller and arranged in a binary tree configuration having stages, for receiving the output data from the processor elements responsive to the synchronization signal generated by and received from the controller, for performing a predetermined calculation based on the stages in the binary tree configuration, and for outputting calculated data to the controller; and

calculation control means, connected between said data collection means and the controller and arranged in series corresponding to each of the stages, for transmitting the calculation control signal from the controller to said data collection means based upon a pipe-line method to perform the predetermined calculation in said data collection means,

wherein said data collection means comprises a plurality of gathering logic units connected to each other in the binary tree configuration having the stages, first gathering logic units of a first stage of the stages receives the output data from each of the processor elements and outputs first calculation data, second gathering logic units of a second stage of the stages receives the first calculation data obtained from the first stage, and the first calculation data obtained from the second stage is output to a final gathering logic unit of a first stage of the stages as second calculation data, the final calculation data obtained from the final gathering logic unit of the final stage responsive to said second calculation data is output to the controller, and

wherein said calculation control means comprises a plurality of control registers each corresponding to one of the stages, each of the plurality of control registers connected in series to each other by the pipe-line method, and each of the plurality of control registers sequentially outputting the calculation control signal to each of the gathering logic units in the corresponding stage.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L7: Entry 3 of 4 File: USPT Nov 14, 2000

DOCUMENT-IDENTIFIER: US RE36954 E
TITLE: SIMD system having logic units arranged in stages of tree structure and operation of stages controlled through respective control registers

Brief Summary Text (5):
Parallel computer systems are widely used, particularly, in the field of CAD (Computer Aided Design) which necessitates high speed calculation for a LSI (large scale integrated) circuit design. Accordingly, it is desirable to improve techniques to make these processor elements operate more efficiently in order to meet the requirements of high density and high speed LSI.

CLAIMS:

1. A parallel computer system using a single instruction stream multiple data stream (SIMD) method, said parallel computer system having a controller and a plurality of processor elements, each of the processor elements having storage means for storing data to be processed and outputting output data processed by the processor elements, the controller controls operation of the processor elements, and said parallel computer system performing processing of the data based on a calculation control signal and a synchronization signal transmitted from the controller, said parallel computer system comprising:

data collection means, connected between the processor elements and the controller and arranged in a .[.binary.]. tree configuration having stages, for receiving the output data from the processor elements responsive to the synchronization signal generated by and received from the controller, for performing a predetermined calculation based on the stages in the .[.binary.]. tree configuration, and for outputting calculated data to the controller; and

calculation control means, connected between said data collection means and the controller and arranged in series corresponding to each of the stages, for transmitting the calculation control signal from the controller to said data collection means based upon a pipe-line method to perform the predetermined calculation in said data collection means,

wherein said data collection means comprises a plurality of gathering logic units connected to each other in the .[.binary.]. tree configuration having the stages, first gathering logic units of a first stage of the stages receives the output data from each of the processor elements and outputs first calculation data, second gathering logic units of a second stage of the stages receives the first calculation data obtained from the first stage, and the first calculation data obtained from the second stage is output to a final gathering logic unit of a first stage of the stages as second calculation data, the final calculation data obtained from the final gathering logic unit of the final stage responsive to said second calculation data is output to the controller, and

wherein said calculation control means comprises a plurality of control registers each corresponding to one of the stages, each of the plurality of control registers connected in series to each other by the pipe-line method, and each of the

plurality of control registers sequentially outputting the calculation control signal to each of the gathering logic units in the corresponding stage.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L9: Entry 8 of 10 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5872988 A
TITLE: Parallel data processing device having a concatenated data path between elementary processors

Detailed Description Text (4):
FIG. 2 shows the architecture of a part of a data processing device in accordance with the invention, said architecture emphasizing interactions between the data streams implemented in and/or by elementary processors operating in the SIMD mode. The plurality of elementary processors EP1-EPn receive at their input instructions from a control bus IC and data from a data bus ID. Each elementary processor delivers a result to an individual output bus OB1-OBn, via output means 17.sub.1 - 17.sub.n. In order to ensure that each elementary processor can communicate with its nearest neighbours, each elementary processor comprises communication means 15.sub.1 -15.sub.n enabling the transfer of data via a concatenated path DP. It is thus possible to transfer data gradually to the other elementary processors. These data exchanges may concern any data present in the elementary processor. The data streams existing in each elementary processor, therefore, are no longer independent. This is particularly useful when the contribution made by an elementary processor is associated with the contributions made by the other processors. This is the case, for example when the plurality of elementary processors EP1-EPn constitutes a vectorial processor VP delivering a plurality of results constituting a vectorial result. The latter is then generally processed by a vector-scalar unit VSU which transforms the vectorial type of result into a scalar type of result.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L9: Entry 9 of 10 File: USPT Jul 21, 1998

DOCUMENT-IDENTIFIER: US 5784076 A
TITLE: Video processor implementing various data translations using control registers

Abstract Text (1):
Advantage is taken of Very Large Scale Integrated (VLSI) circuit design and manufacture to provide, in a digital data handling system handling display signal streams, a video processor which is capable of high performance due to vector processing and special addressing modes. A single VLSI device has a plurality of processors which cooperate for generating video signal streams and which employ distinctive addressing modes for memory elements of the device. Each of the plurality of processors has associated instruction and data caches, and the processors are joined together by a wide data bus formed on the same substrate as the processors. Each processor has a load/store unit, a translation unit associated with the load/store unit, and an index control register for controlling any translation of data bit streams passed through the load/store unit. The presence and operation of the translation unit enables the device to efficiently process data streams of varying types, thereby broadening the applicability of the device.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L8: Entry 2 of 5 File: USPT Jan 9, 2001

DOCUMENT-IDENTIFIER: US 6172990 B1
TITLE: Media access control micro-RISC stream processor and method for implementing the same

Parent Case Text (2):
This application claims the benefit of U.S. Provisional Patent Application having Ser. No. 60/050,210, filed on Jun. 19, 1997, entitled "MEDIA ACCESS CONTROL MICRO-RISC STREAM PROCESSOR AND METHOD FOR IMPLEMENTING THE SAME." This application is hereby incorporated by reference.

Brief Summary Text (3):
The present invention relates generally to integrated circuit devices used for processing data through communication networks, and more particularly, to methods and apparatuses for high speed packet processing within the media access control level to reduce host central processing unit processing loads.

Brief Summary Text (8):
Above the physical layer 12, a data link layer 14 is defined for providing reliable transmission of data over a network while performing appropriate interfacing with physical layer 12 and a network layer 16. As shown, data link layer 14 generally includes a logical link layer (LLC) 14a and a media access control layer (MAC) 14b. LLC layer 14a is generally a software function that is responsible for attaching control information to the data being transmitted from network layer 16 to MAC layer 14b. On the other hand, MAC layer 14b is responsible for scheduling, transmitting and receiving data over a link. Thus, MAC layer 14b is primarily responsible for controlling the flow of data over a network, ensuring that transmission errors are detected, and ensuring that transmissions are appropriately synchronized. As is well known in the art, MAC layer 14b generally schedules and controls the access of data to physical layer 12 using a well known carrier sense multiple access with collision detection (CSMA/CD) algorithm.

Brief Summary Text (14):
As an example, when packet data is received by the MAC layer 14b from the lower physical layer 12, the CPU is conventionally required to scan through each and every bit of data in the order received to locate the byte location of headers and data that may be of interest to upper layer protocols. Once the CPU has laboriously searched the entire packet and ascertained the particular byte locations of interest in each packet, all of this information is made available to upper layers, such as the network layer 16, transport layer 18, session layer 20, presentation layer 22 or the application layer 24. Once these upper layers have the information they need regarding the received packet, these upper layers will be able to complete their assigned tasks. Unfortunately, in high speed networks, the demands on a host CPU tends to increase to levels where scanning through each byte of each received packets is no longer possible without introducing decoding, transmission or packet routing delays.

Brief Summary Text (16):
In view of the foregoing, there is a need for methods and apparatuses for media access control layer processing that is well suited to increase transmit and

receive packet processing rates while reducing a host CPU's processing burden.

Brief Summary Text (21):
In yet another embodiment, a packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer is disclosed. The packet data processor includes a memory configured to receive executable microcode defining a type of data structure to be built from the received packet data. A pipeline register stage having a plurality of registers for sequentially receiving and temporarily storing words of the received packet data and, each of the plurality of registers in the pipeline register stage are coupled to a pipeline multiplexor that is capable of reading a portion of the words that are temporarily stored in the pipeline register stage. The packet data processor further includes an analyzing computer that is configured to examine the received packet data output from the pipeline multiplexor, and store a element of the received packet data generated by the analyzing computer into a register file. Further, the packet data processor includes an execution logic unit that is configured to receive the executable microcode from the memory. The execution logic unit is preferably designed to control the examination of the received packet by the analyzing computer.

Detailed Description Text (2):
An invention is described for a high speed media access control layer micro-Risc engine that is user programmable to process packet data in-line while streaming packets in or out of the media access control layer core. Also disclosed are methods for user programmable in-line processing (e.g., parsing) of receive and transmit packet data while performing high speed streaming of packet data in or out of the media access control layer. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be obvious, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

Detailed Description Text (3):
1. Media Access Control Architecture

Detailed Description Text (4):
FIG. 2A is an architectural diagram of a flow based media access controller (MAC) 150 for high speed transmissions in accordance with one embodiment of the present invention. In a preferred embodiment, gigabit speed or greater Ethernet transmissions are contemplated. However, it should be appreciated that the architecture is equally applicable to other transmission protocols and both higher and lower speed transmissions. In one embodiment, flow based MAC 150 is a parallel data and control processing architecture. As illustrated in FIG. 2, the flow based MAC 150 interfaces with a network data system bus 101 where both data and control information are processed, and a management/control bus 102 where both control and management data are passed. As data is passed through network data system bus 101 and processed through the various processing blocks of flow based MAC 150, control information may also be simultaneously passed through network data system bus 101. It is important to realize that this type of parallel processing provides the ability to change the processing parameters within flow based MAC 150 at any given time (i.e., even while packet data is being processed).

Detailed Description Text (5):
By way of example, suppose data is being received from an upper LLC layer, and is being processed through various processing blocks where a preamble field and a CRC field are appended to form a packet. Due to the parallel processing nature of flow based MAC 150, control information may be simultaneously passed through network data system bus 101 to modify portions of the packet that has not yet been processed. Accordingly, the parallel processing nature of flow based MAC 150 is

capable of passing appropriate control information to alter specific processing parameters even while data is currently being processed.

Detailed Description Text (6):
Referring first to the transmission side, when data is initially received from the upper LLC layer through network data system bus 101, data is transferred to a network data bus interface controller (BIC) 104. In this embodiment, network data BIC 104 may be any suitable interface controller such as a slave interface and a direct memory access (DMA) on-board interface. As shown, a first data/control path 144a and a second data/control path 144b may be used to interconnect network data bus interface 101 to network data BIC 104 when high performance switching tasks are required of flow based MAC 150. By way of example, first data/control path 144a may be used for performing transfers from the upper LLC layer to flow based MAC 150, and second data/control path 144b may be used for performing transfers from flow based MAC 150 to the upper LLC layer. Of course, it is also contemplated that a single bidirectional data/control path may be used by combining 144a and 144b to perform the aforementioned control and data transfers.

Detailed Description Text (9):
In yet another embodiment, control information may be embedded within packets buffered in FIFO Tx 106. In this manner, the processing parameters may be modifiable in a pipe-lined packet-by-packet basis. By way of example, the embedded control information may contain modifications to the processing parameters as well as identifying information to single out a particular packet for modification. It should be appreciated that having a smart network flow managing FIFO Tx controller 110 also facilities network management and associated testing protocols. Although few circumstances should require the processing parameters to be changed for each successive packet, it will be appreciated that the ability to modify the processing parameters for any given packet in a packet stream is a powerful feature.

Detailed Description Text (10):
Once network flow managing FIFO Tx controller 110 performs any requested processing based on received control information, micro-RISC stream processor 114a is suited to performs various user programmable packet processing, parsing, filtering and encapsulation operations. By way of example, micro-RISC stream processor 114a operates in an in-line manner for modifying data stream characteristics. Preferably, micro-RISC stream processor 114a (as well as 114b for the receive side) operates on 32 bit-word portions at one time to efficiently process information along flow based MAC 150, while achieving giga-bit speed (and higher) performance. Furthermore, instructions are preferably triggered off of the byte stream. In this embodiment, micro-RISC stream processor 114a is also suited to operate in various addressing modes such as, for example, relative byte count mode.

Detailed Description Text (11):
Internally, micro-RISC stream processor 114a will preferably have a set of general purpose registers, data structure registers, and analyzing computer units. By way of example, the analyzing computer units may include a CRC unit, a compressed hash data unit, an ALU unit, a programmable checksum generator, a CAM, and comparators. Further, micro-RISC stream processor 114a is preferably capable of operating in a conditional, branch, and loop mode, which provides additional flexibility and improved performance. Finally, micro-RISC stream processor 114a processing instructions may include a number of inventive packet field manipulations. Exemplary manipulations may include: CUT, CLEAR, COPY, APPEND, INSERT, AND, OR, XOR, MOVE, JUMP for specialized header generation, separating data and headers, IP_CHKSUM checking and length calculation.

Detailed Description Text (13):
Still referring to FIG. 2A, once appropriate data and control information is processed within micro-RISC stream processor 114a, data is transferred via a data path 113a to a SUPERMAC Tx controller 118 which is preferably a state machine

configured to process packets received from micro-RISC stream processor 114a and output the processed packets to a physical (PHY) medium 140. For more information on the SUPERMAC Tx controller 118 and the SUPERMAC Rx controller 120 of FIG. 2A, reference may be made to the following related U.S. patent applications: (1) Ser. No. 09596745 (attorney docket no. XAQTP001A), entitled "Media Access Control Architectures And Network Management Systems," (2) Ser. No. 08845563 (attorney docket no. XAQTP002), entitled "Media Access Control Transmitter And Network Management System," and (3) Ser. No. 08845272 (attorney docket no. XAQTP003), entitled "Media Access Control Receiver And Network Management System." These applications are hereby incorporated by reference.

Detailed Description Text (14):
FIG. 2A also shows a SUPERMAC management block 117 that is responsible for interfacing between transmitting SUPERMAC Tx controller 118 and a receiving SUPERMAC Rx controller 120. SUPERMAC management block 117 also interfaces with network flow managing FIFO Tx controller 110, a network flow managing FIFO Rx controller 112, and network data BIC 104. Generally, SUPERMAC management block 117 functions as an interface that receives flow control information, auto negotiation commands, physical management commands, and pause frame information (i.e., pause frames are used by a receiving unit to notify a transmitting unit to cease the transmission of data until a receiving buffer is free).

Detailed Description Text (17):
As in the transmitting side, events performed in SUPERMAC Rx controller 120, and micro-RISC stream processor 114b are both linked to micro-RISC stream processor 114c, which accounts for those events in statistical counters 128. Preferably, network flow managing FIFO Rx controller 112 is capable of assigning a number to each of the packets received by FIFO Rx 108. Because FIFO Rx controller 112 is knowledgeable of the numbers assigned to each packet, a control signal may be transferred to FIFO Rx controller 112 requesting that a particular numbered packet stored in FIFO Rx 108 be transferred (i.e., to LLC layer or PEP 124 for management purposes). Once data is transferred out of multi-packet queue FIFO Rx 108 and into network data BIC 104, in a switched environment, data is passed through data path 144b onto network data system bus 101. Of course, a single bidirectional data path may alternatively be used in place of paths 144a and 144b.

Detailed Description Text (19):
In this embodiment, streaming BIC 122 is preferably implemented for passing control information and performing data management tasks. By way of example, in performing a network management task, it may be necessary to pull (i.e., filter) a particular packet of information from the path of packets being processed through network data BIC 104. Once the desired packet is identified, it may then be filtered by micro-RISC stream processor 114c that lies within parallel event processor (PEP) 124.

Detailed Description Text (22):
Accordingly, PEP 124 includes an inventive packet buffer 125 for storing appropriate packets that are implemented by the network management protocols such as SNMP and RMON. By way of example, if a user wants to monitor certain packets within the data stream being processed through network data BIC 104, the micro-RISC stream processors 114b and 114c will filter out the desired packets that are subsequently stored in packet buffer 125. Also included within PEP 124 is command and status registers 126, such that the command registers receive associated control signals from management/control bus 102 through streaming control BIC 122. In one embodiment, 114b and 114c may be the same processing entity.

Detailed Description Text (27):
As described above, the user may program micro-RISC stream processor 114b to generate custom data structures containing, for example, a pointer to the start of an internet protocol (IP) header, a pointer to the start of a transmission control protocol (TCP) header, and a pointer to the start of a simple mail transfer

protocol (SMTP) header. Other exemplary data structures may be programmed to include portions of the packet data itself such as an IP destinations address and compressed hashed data. Because the type of data structure and the content of the data structures are fully programmable by the user, other exemplary data structures may include "flag" data structures where each bit is programmed to identify one protocol or another, or "field" data structures where multiple bits are coded to identify different networking protocols.

Detailed Description Text (29):
By way of example, the structural separation provides a built-in "delay" to facilitate processing, such as, packet header modifications, and protocol translation processing. In one embodiment, packet header modification may include packet fragmentation and transformation to convert standard Ethernet packets into an asynchronous transfer mode (ATM) cells. Other packet translation functionalities may include translational bridging between Ethernet, Token Ring and FDDI. The split structure also allows a user to program micro-RISC stream processor 114b to perform a checksum operation after being output from the multi-packet queue FIFO Rx 108.

Detailed Description Text (39):
When this happens, the selected word of the in-coming packet has now been stored in stage 1324 of the pipeline register stages 323. At the same time, the microcode contained within execution instruction register 306 is passed to execution logic 312 which controls the current action of the micro-RISC stream processor 114b. In one embodiment, execution logic 312 communicates with a MUX 320, a MUX 318, a MUX 314, a CRC unit 330, a HASH 331, an arithmetic logic unit (ALU) 332, CAM 334, comparators 336, and a programmable checksum generator 333. As shown, CRC 330, HASH 331, ALU 332, CAM 334, comparators 336, and a programmable checksum generator 333 are part of an analyzing computer 337 that is configured to act on the word of interest (of a current packet) identified by word count 308. As described above, if SUPERMAC Rx controller 120 passes the received packet along with the CRC field, CRC 330 is preferably configured to perform a CRC calculation and strip the CRC field before the packet is transferred to the upper LLC layer. In one embodiment, the CRC calculation is a 32 bit or 16 bit cyclic redundancy check using a generator polynomial.

Detailed Description Text (41):
If the user desired to create a data structure having a pointer to the currently selected 32-bit word (which may be the start of a header), then the word count will be transferred from word counter 307 to MUX 318 to be input into a current data structure. In this embodiment, the current data structure will preferably be stored in a data structure register file 316. Once all of the data of interest for a current packet has been parsed and stored into the data structure register file 316, the data structure will be appended to the beginning of the packet data being output from MUX 314.

Detailed Description Text (42):
In another example, if the user desires to construct a data structure that includes hashed data (i.e., compressed packet data), the hashed data will be processed in HASH 331 and then passed to MUX 318. Still further, the user may desire that portions (i.e., selected 32-bit words) of the received packet data be placed into the data structure for quick reference by upper layer protocols. Once an entry is made into the current data structure, for the current packet, CAM 334 and comparators 336 generate comparison control signals that are transferred to an encoder 317 and to a next address logic 310. The control information provided to encoder 317 is preferably used to set (i.e., through encoded set bits) the type of data structure the user wants to build, and the control information provided to the next address logic is used to identify the next address from the microcode stored in RAM 302. Thus, based on the comparisons performed in CAM 334 and comparators 336, a branch operation or a move operation will be performed. By way of example, when a branch operation is performed, the next address logic 310 will locate

another address location in RAM 302. Further, if a move operation is performed, one of the analyzing computer 337 units will transfer an output to MUX 318 and into data structure register file 316.

Detailed Description Text (45):
Once the next location within the microcode contained within RAM 302 is ascertained, that portion of the microcode is again transferred to instruction register 304 and word count 308. Again, word count 308 will contain the next word count of interest within the current packet being received. By way of example, since the last word of interest was word 57, the next exemplary word of interest may be word 88. In this example, word 88 may identify the beginning of a header, the beginning of data to be compressed (e.g., hashed) or the beginning of data to be captured. When word counter 307 reaches the 88th word in the packet, the microcode stored in instruction register 304 is shifted into execution register 306 to enable the executing on the newly received data word that is currently stored in stage 1324 of the pipeline register stage 323.

Detailed Description Text (48):
As mentioned earlier, when the upper layers receive the packets having the appended data structures, the upper layers may simply read the data they need to complete packet routing without having to examine substantially all of the received packet to locate the information that is of interest to the upper layers. It should be noted that most of the time, each packet may have similar header information (i.e., the IP header) located in different byte locations within a received packet, and therefore, a host's CPU is typically required to laboriously scan through most of the contents of each packet before it can perform any necessary routing or processing. Accordingly, by appending a user defined data structure to the front of a received packet, even greater than gigabit Ethernet transmission speeds may be attained with substantially fewer CPU interrupts.

Detailed Description Text (51):
Once the initial skip has been performed into the received packet, the method will proceed to a step 406 where the received packet is examined by an analyzing computer contained in the micro-RISC stream processor 114b in accordance with the user defined processing instructions provided in the user defined microcode. By way of example, the packet is typically examined to ascertain the word count location of a particular header, or a particular piece of data. Once the analyzing computer computes the desired data to be appended to a data structure, the desired data is passed to a multiplexor.

Detailed Description Text (52):
After the packet has been examined in step 406, the method will proceed to step 408 where the identified pointer, data, or hashed data (or a combination thereof) is stored into the defined data structure. The method will then proceed to a decision step 410 where it is determined if there are any more positions of interest in the examined packet. By way of example, if the microcode determines that there are five separate headers locations of interest, then the method will proceed to a step 412 where micro-RISC stream processor 114a will skip to new position in the received packet in response to the examination of the received packet. In this embodiment, the new skip location will preferably be the location ascertained by the microcode address selected by a next address logic unit contained within micro-RISC stream processor 114b.

Detailed Description Text (62):
As described above, the analyzing computer preferably contains a CRC unit, a hash unit, an ALU unit, a CAM unit, comparators, an encryption/decryption unit, a compression/decompression unit, and a programmable checksum generator unit. The analyzing computer preferably receives data from a pipeline register stage and then acts upon that 32-bit word data based on control information provided by an execution logic unit. Once the desired 32-bit word has been loaded into the

analyzing unit from the pipeline register stage in step 442, the method will proceed to a step 444 where the microcode contained in an execution instruction register is executed by the execution logic to control the processing for the current 32-bit word.

Detailed Description Text (65):
FIG. 4D is a more detailed flowchart describing the processing performed during a skip through the received packet data performed in step 412 of FIG. 4A in accordance with one embodiment of the present invention. The method begins at step 460 where a word count provided by the microcode provides the micro-RISC stream processor 114a with an amount of bytes to skip forward in the received packet. By way of example, if it is determined that word 52 contains an IP header, and the IP header is a desired position for which the user wants to generate a pointer for the data structure, the micro-RISC stream processor 114a will allow all 32-bit words received up to word 51 to pass before being directed by the execution logic to place a pointer into the data structure for the 52nd word.

Detailed Description Text (67):
FIGS. 5A through 5D show exemplary data structures which may be programmably created for packets being received by micro-RISC stream processor 114b in accordance with one embodiment of the present invention. As shown in FIG. 5A, a user may program a data structure for a packet A to include a pointer to the start of an IP header, a pointer to the start of a TCP header, a pointer to the start of an SMTP header, a pointer to the start of an application header and a data portion. Further, the data structure may include actual packet portions, such as, an IP source address, an IP destination address, a source and destination port number, and hashed data.

Detailed Description Text (68):
FIG. 5B shows a data structure for a packet B in which only pointers to specific portions of packet B are identified. As such, the user may have a pointer directed to the IP header, a TCP header, an SMTP header and so on. In another example, FIG. 5C shows a data structure having only packet data portions, such as, an IP source address, an IP destination address and a source and destination port number without the inclusion of pointers or hash data. In still another example, FIG. 5D shows a data structure for a packet D that may be programmed to include only hashed data. As is well known in the art, hashed data refers to compression data, which is typically done to reduce processing loads on host CPUs.

Detailed Description Text (69):
FIG. 5E shows packets A through D having an associated data structure appended to the front of each packet in accordance with one embodiment of the present invention. In general, by providing the information that is of interest to upper layer protocols at the front of the packet, in compact data structure arrangement, a host CPU is no longer required to laboriously scan and search through substantially the whole packet in order to ascertain the location of headers, or data of interest.

Detailed Description Text (70):
FIG. 6A shows a data structure for a packet F which has been programmed by a user to include a plurality of flags in accordance with one embodiment of the present invention. In one embodiment, the flag data structure is preferably composed of 32 bits which may be used to identify the type of packet or its associated layer protocol. In the example provided, the first flag may be used to determine whether the incoming packet is an IP or an internet packet exchange (IPX) protocol, the second flag may be used to distinguish between a TCP or a user datagram protocol (UDP) protocol, and the third bit may be used to distinguish between a TELNET (i.e., TELNET is a TCP/IP application that enables a user to log in to a remote device) or an SMTP protocol. In another example, the last exemplary flag of the data structure may be used to determine if the packet is an ICMP (i.e., internet

control message protocol) or not an ICMP packet.

Detailed Description Text (75):
In a preferred embodiment, micro-RISC stream processor 114a as shown in FIG. 2A includes a number of advantageous packet data processing capabilities. For example, micro-RISC stream processor 114a is preferably well suited to encapsulate out-going packets with various types of headers to improve packet switching, routing, or convert Ethernet packets into ATM cells. It is important to keep in mind that the transmitter micro-RISC stream processor 114a is also configured to process and parse the packet data in line with the streaming out of the packet data, thereby advantageously avoiding latencies in transmissions. Further, because the packet encapsulation and translation operations are completed in the micro-RISC stream processor 114a which lies within flow based MAC 150, a substantial processing burden is lifted from a host CPU, which is advantageously freed from the laborious task of parsing the out-going packet to determine appropriate translations or encapsulations.

Detailed Description Text (77):
At this point, packet 802 and appended index 804 is passed into micro-RISC stream processor 114a where the packet data may be processed for transmission out to a remote host (i.e., switch, router, hub, etc.) through the physical layer 140, as shown in FIG. 2A. In this example, the switch table lookup 806 may also append a command header 805 that enables micro-RISC stream processor 114a to determine what type of processing to perform on the packet 802. Once the micro-RISC stream processor 114a uses the command header 805 to create the desired encapsulation headers, the command header 805 will no longer be used. In this embodiment, micro-RISC stream processor 114a is also well suited to attach an encapsulation header 808 at the front of appended index 804 and packet data 802. In addition, micro-RISC stream processor 114a may be programmed to calculate a new cyclic redundancy check (CRC), which may be appended to the back of packet 802 before being transmitted to the remote host.

Detailed Description Text (78):
In one embodiment, encapsulation header 808 may be a virtual local area network (VLAN) header, which is well known to assist networks in filtering traffic based upon complex schemes other than source and destination addresses. In another example, condition based filtering may also be performed by the micro-RISC stream processors. Thus, the VLAN header could empower a network to perform efficient routing based on Ethernet addresses, IP network numbers, or special VLAN designators. In still another embodiment, micro-RISC stream processor 114a may be configured to perform Cisco inter-switch link (ISL) tagging schemes via encapsulation header 808.

Detailed Description Text (79):
FIG. 9 illustrates a number of functionalities that may be performed within micro-RISC stream processor 114a in accordance with one embodiment of the present invention. As described above, micro-RISC stream processor 114a may be well suited to provide an encapsulation header 808a, which may be an ISL header used in well known Cisco tagging schemes. Also shown is an original CRC 810a field that may be calculated by micro-RISC stream processor 114a, and appended to out-going packets (in-line). Also shown is an additional new CRC 811 which may be appended to the out-going packet in SuperMAC controller Tx 118 before being transmitted to the physical medium 140, as described above.

Detailed Description Text (80):
In the next example, micro-RISC stream processor 114a may be well suited to perform ATM cell fragmentation and reassembly tasks. By way of example, when micro-RISC stream processor 114a receives an original Ethernet packet containing a destination address (DA) 902, a source address (SA) 904, and data 906, a fragmentation and reassemble operation may be performed on data 906, source address 904, and

destination address 902. Once fragmented, an ATM header 808b may be appended to the front of the ATM cell. Because ATM cells are generally of a fixed size, the remaining data that was not appended to the first out-going ATM cell, will be appended to the following ATM cell which is also assembled with its own ATM header 808b.

Detailed Description Text (81):
In still another example, micro-RISC stream processor 114b may be well suited for performing IP switching, wherein an IP header 910 is parsed and indexed with an IP index 912, which is appended to the front of the packet. In this embodiment, micro-RISC stream processor 114a is preferably well suited to generate IP index 912 and compress it through a suitable hashing operation. Therefore, for IP switching, the index is a combination of the source and destination ports and the source and destination MAC addresses, and in some cases, parts of the IP header itself. In this manner, a small width index (12 bits for IP switching), may be used to tag all frames and to switch frames. As a further embodiment, the micro-RISC stream processors of the described embodiments may also be useful in performing IP fragmentation and IP reassembly to reduce the load on a host's CPU. In yet another embodiment, IP checksum functions may also be performed within the various embodiments of the micro-RISC stream processors described herein.

Detailed Description Text (83):
The present invention may be implemented using any type of integrated circuit logic or software driven computer-implemented operations. By way of example, a hardware description language (HDL) based design and synthesis program may be used to design the silicon-level circuitry necessary to appropriately perform the data and control operations in accordance with one embodiment of the present invention. By way of example, a VHDL.RTM. hardware description language based on standard available from IEEE of New York, N.Y. may be used to design an appropriate silicon-level layout. Although any suitable design tool may be used, another layout tool may include a hardware description language "Verilog.RTM." tool available from Cadence Design Systems, Inc. of Santa Clara, Calif.

Detailed Description Text (87):
The microprocessor 1016 is a general purpose digital processor which controls the operation of the computer system 1000. The microprocessor 1016 can be a single-chip processor or can be implemented with multiple components. Using instructions retrieved from memory, the microprocessor 1016 controls the reception and manipulation of input data and the output and display of data on output devices. According to the invention, a particular function of microprocessor 1016 is to assist in the packet processing and network management tasks.

CLAIMS:

10. A method for processing packet data received from a physical layer as recited in claim 1, wherein the packet data processing is performed by a media access controller layer that communicates with a logic link control layer.

21. A method for processing packet data received from a lower layer as recited in claim 17, wherein the element is a pointer to a header to be stored in the pointer data structure.

24. A method for processing packet data received from a lower layer as recited in claim 13, wherein the packet data processing is performed by a media access controller layer that communicates with a logic link control layer.

27. A packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer, comprising:

a memory configured to receive executable microcode defining a type of data

structure to be built from the received packet data;

a pipeline register stage having a plurality of registers for sequentially receiving and temporarily storing words of the received packet data, each of the plurality of registers in the pipeline register stage being coupled to a pipeline multiplexor capable of reading a portion of the words temporarily stored in the pipeline register stage;

an analyzing computer configured to examine the received packet data output from the pipeline multiplexor, and storing a element of the received packet data generated by the analyzing computer into a register file; and

an execution logic unit configured to receive the executable microcode from the memory, the execution logic unit being designed to control the examination of the received packet by the analyzing computer.

38. A packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer as recited in claim 34, wherein the element is a pointer to a header to be stored in the pointer data structure.

40. A packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer as recited in claim 27, wherein the analyzing computer includes a CRC module for performing a CRC check and stripping a CRC field from the received packet data in response to control signals received from the execution logic unit.

41. A packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer as recited in claim 27, wherein the analyzing computer includes a hashing module for compressing at least a part of the received packet data in response to control signals received from the execution logic unit.

42. A packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer as recited in claim 27, wherein the analyzing computer includes a programmable checksum generator for performing polynomial divider operations in response to control signals received from the execution logic unit.

43. A packet data processor for parsing received packet data in-line with streaming the packet data to an upper layer as recited in claim 27, wherein the analyzing computer includes a content addressed memory unit for performing value comparisons in response to control signals received from the execution logic unit.

44. In a packet data processor for processing packet data in-line with the streaming of the packet data to a transmit media access controller that is configured to transmit the packet data over a network link, a method comprising:

identifying a packet to be transmitted over the network link;

generating a tag header;

generating a cyclic redundancy check header (CRC); and

appending the cyclic redundancy check header and the tag header to the packet that was identified to be transmitted over the network link before being streamed to the transmit media access controller,

wherein at least one selected from a group consisting of identifying the packet, generating the tag header, generating the CRC header, and appending the CRC header is performed in-line while packet data is being streamed through the processor.

45. The method as recited in claim 44, wherein the tag header is a VLAN header.

46. The method as recited in claim 45, wherein the cyclic redundancy check header is generated for the packet data including the tag header.

48. The method as recited in claim 47, further comprising:

replacing of media access control destination and source fields, the replacement being selectively performed to enable IP fowarding.

49. A method reconstructing packet data in-line with the streaming of the packet data to a physical media, comprising:

fragmenting an Ethernet packet into a plurality of sub-packets;

reassembling the plurality of sub-packets of the Ethernet packet into a plurality of ATM cells; and

appending an ATM header to each of the plurality of ATM cells before being streamed to the physical media,

wherein at least one selected from a group consisting of fragmenting the Ethernet packet, reassembling the plurality of sub-packets of the Ethernet packet, appending the ATM header to each of the plurality of ATM cells is performed in-line while packet data is being streamed to the physical media.

replacing of media access control destination and source fields, the replacement being selectively performed to enable IP forwarding.

Previous Doc Next Doc Go to Doc#